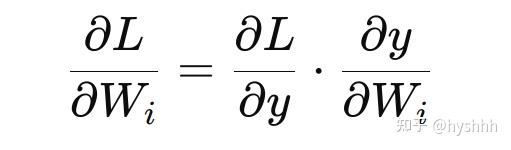
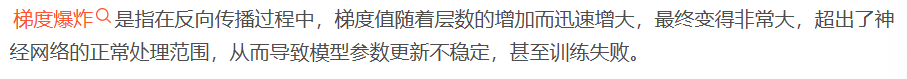
1. Resnet18的实现与分类任务的完成
2. 残差网络为什么应运而生：

在ResNet出现之前，深度学习领域，大家公认一个真理：网络越深，性能越好。道理很简单，层数越多，模型能学到的特征就越复杂、越抽象，识别能力应该越强。大家开始疯狂地堆叠层数，从几层到十几层，再到二三十层。效果也确实不错，成绩一路飙升。但是，网络做得**更深后，**一个反直觉的现象发生了：更深网络的识别准确率，不但没有提升，反而显著下降了。因为在普通深度网络中，随着层数增加，梯度可能会消失或爆炸，使得训练变得困难。

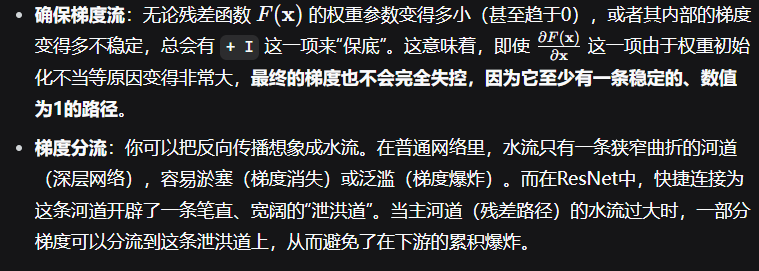
计算梯度公式：

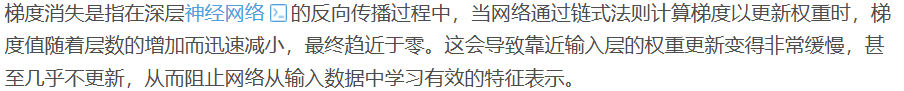




在深层神经网络中，由于链式法则的应用，梯度需要通过多层进行反向传播。如果每一层的梯度都稍微增大一点，那么经过多层传播后，梯度值就会变得非常大，导致梯度爆炸。

我有一点难以理解的是，为什么跳跃连接不会促进梯度爆炸,毕竟，跳跃连接反而使因子更大，ai的解释如下：

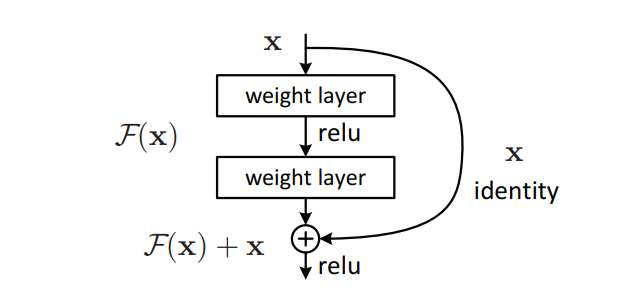




随着网络层数的增加，梯度需要通过更多的层进行反向传播。每一层都可能对梯度进行一定的衰减，因此层数越多，梯度消失的风险就越大。

1. resnet如何解决上诉梯度问题：

resnet本质上还是一种cnn的架构，但是它通过引入“残差学习”来解决深度网络训练中的退化问题。通过绕过某些中间层级，将层的激活值直接链接到后续层。

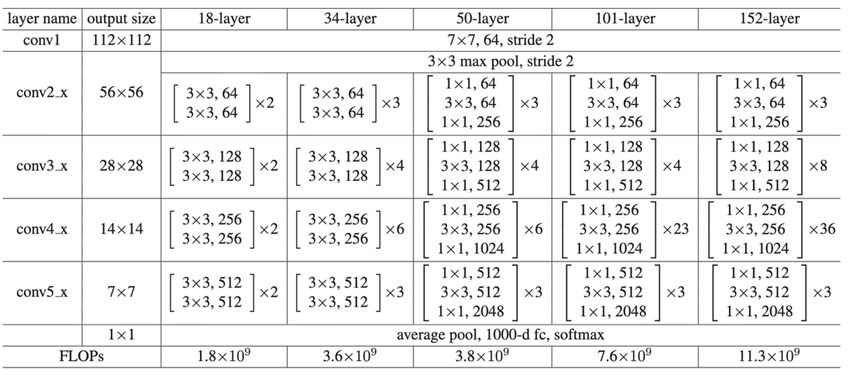


残差网络背后的策略是让网络去拟合残差映射，而不是让层去学习底层的映射。因此，网络不是去拟合比如H(x)这样的初始映射，而是去拟合H(x) - x这样的残差映射。

梯度直接通过残差连接进行传递，使得梯度可以更轻松地传播到前面层。  
反向传播时，跳跃连接保证了梯度可以有效传递到早期层，避免了梯度消失

3.resnet18的实现思路和实现过程：

resnet18的18来自于卷积层和全连接层加起来（不计入池化层）共有18层这个事实：

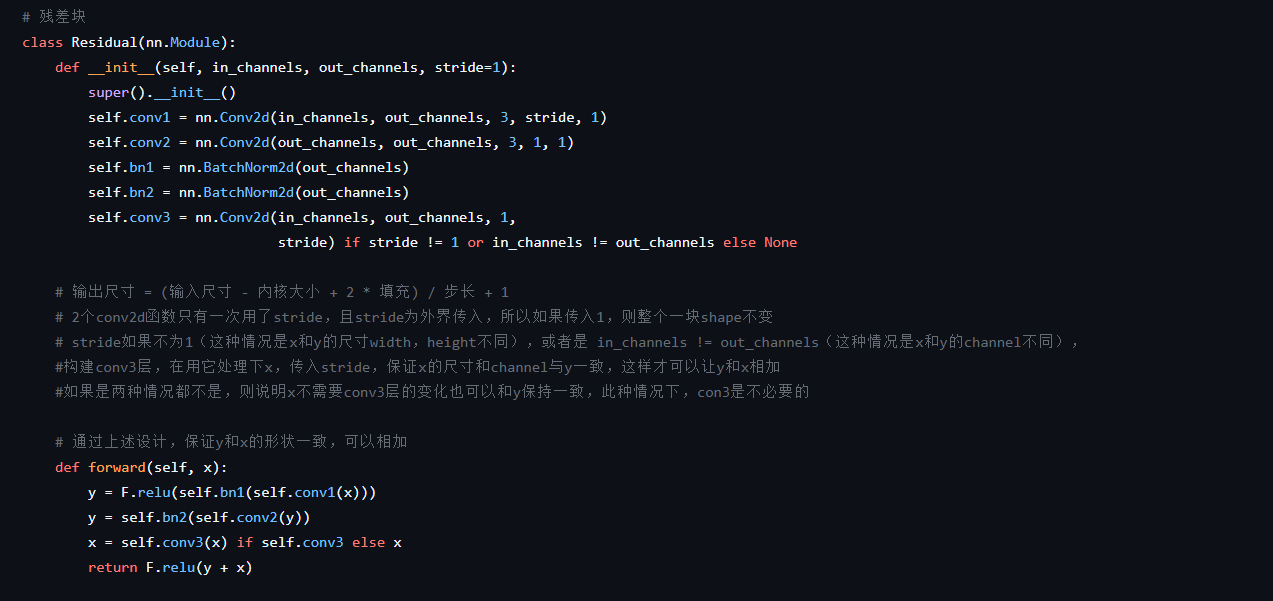
如图初始卷积层：kernel\_size=7,out\_channel=64

Convi\_x(i=2,3,4,5)每一层都各有2个残差块，而每个残差块又有2个卷积层，共有2\*2\*4=16

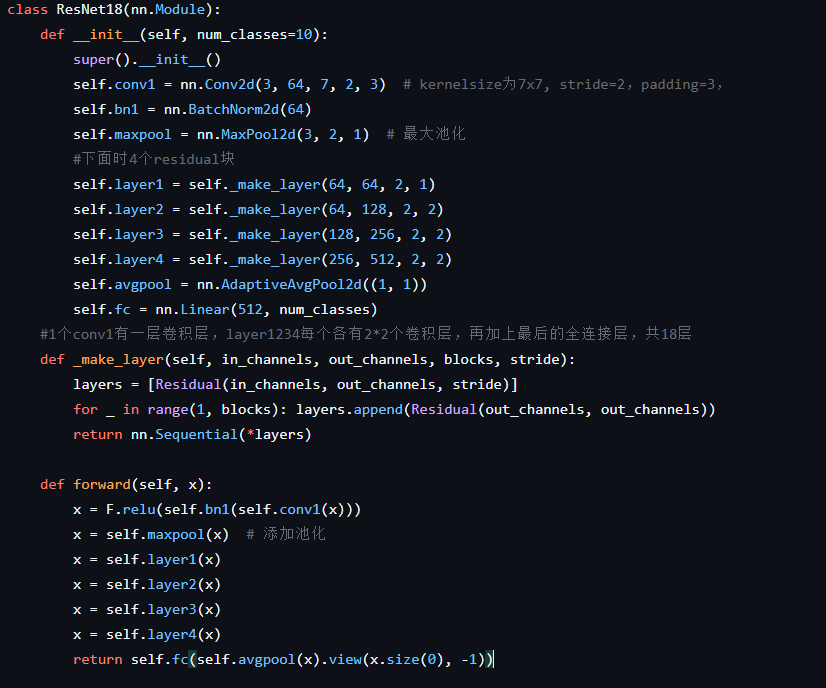
全连接层1层

所以总计18层

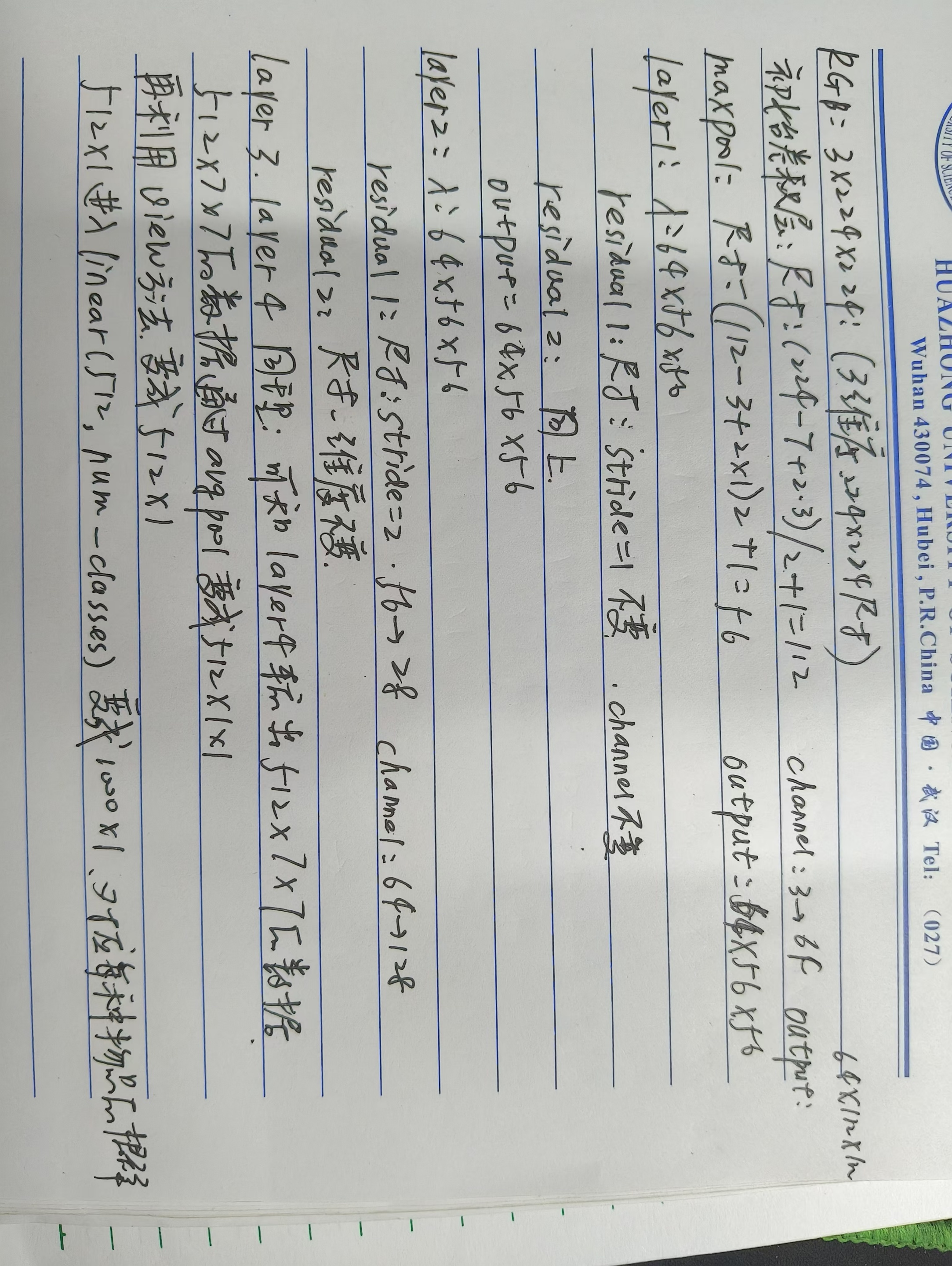
考虑先实现一个residual类，再运用这个类实现resnet18，代码如图：



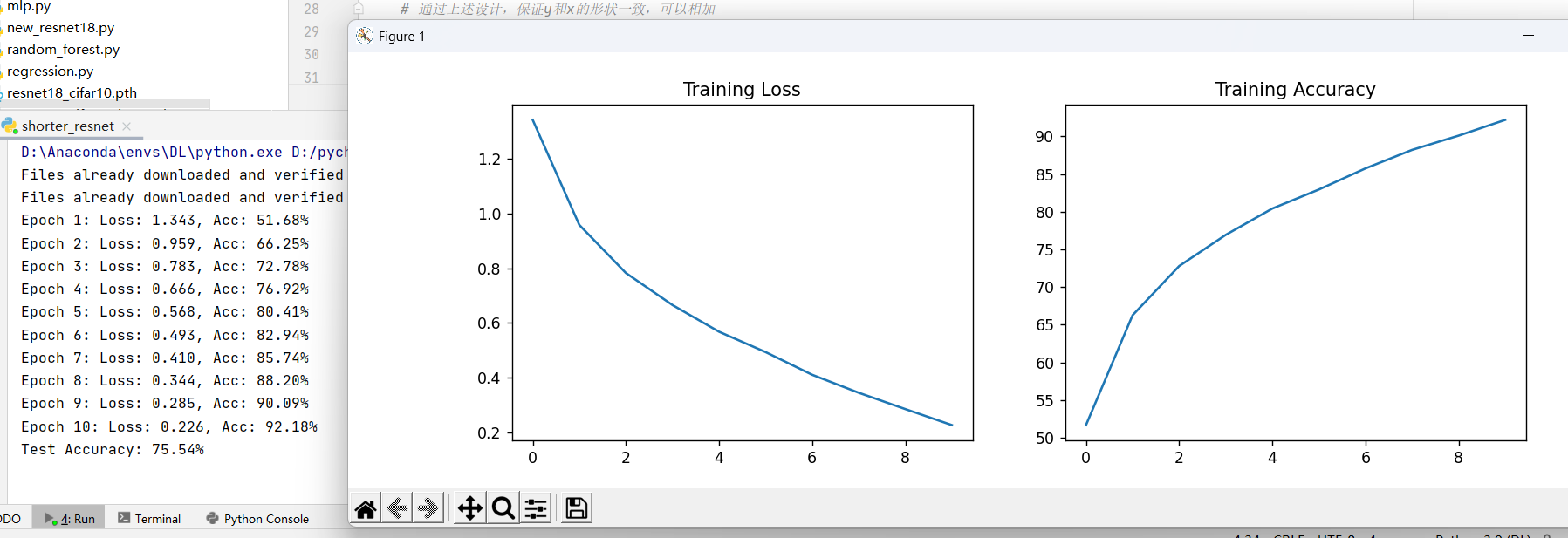
这里有一个细节，就是如何保证y和x可以相加，需要使用conv3层，具体方法在注释中已经详细说明；还有就是从输出尺寸的计算公式，可以知道，kernel size为1时，输入数据的尺寸时不会改变的，这点在后面的使用residual构建resnets18也会体现，即每个residual的第二个残差块不会改变尺寸。



为了更好理解，可以输入一个3\*224\*224的rgb图像来探究它在resnet18的变化：如图：



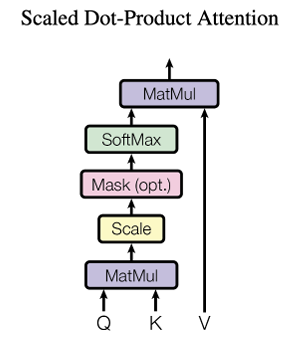
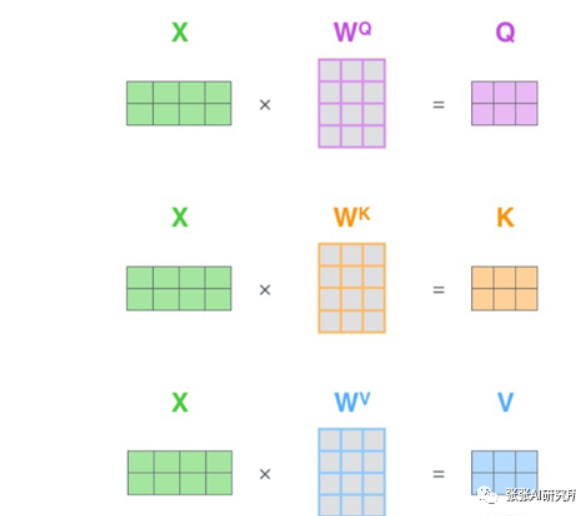
利用resnet18进行分类任务的结果如下：



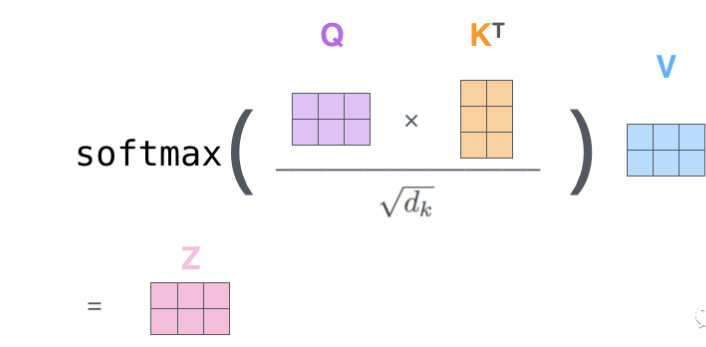
可以看到，resnet18可以有效地解决分类任务，而且准确率也比较高

1. Transformer架构：
2. SelfAttention机制：

将单个序列的不同位置关联起来以计算同一序列的表示的注意机制。Query，key，value由一个数据经过线性变换而来，自注意力机制和注意力机制的区别就在于，注意力机制的查询和键是不同来源的，例如，在Encoder-Decoder模型中，key是Encoder中的元素，而query是Decoder中的元素。

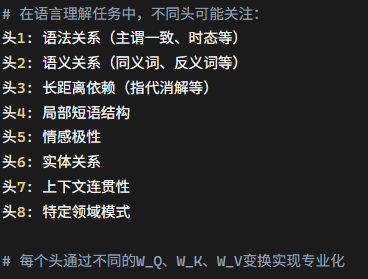
 

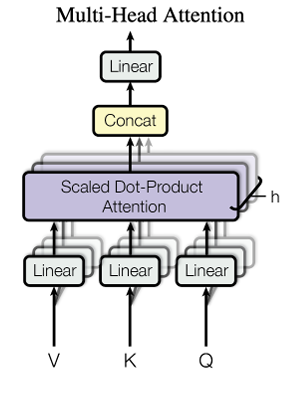
使用缩放点积注意力来评分



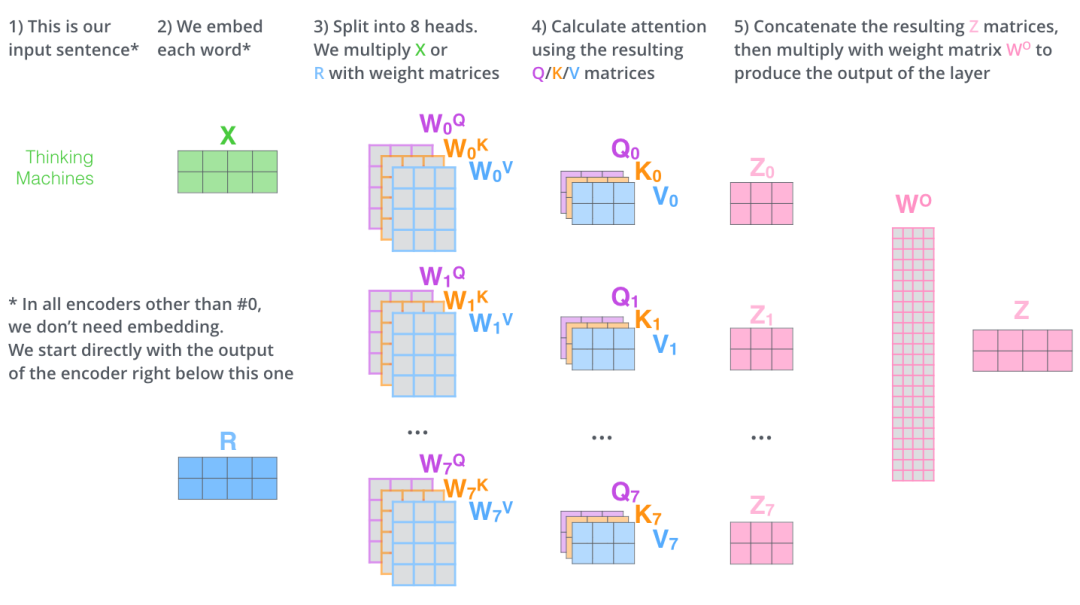
1. 多头注意力：

多头注意力机制的多头表示对每个Query和所有的Key-Value做多次注意力机制。做两次，就是两头，做三次，就是三头。这样做的意义在于获取每个Query和所有的Key-Value的不同的依赖关系。

每个头负责不同的功能有不同的侧重点。



多头注意力实现具体如图：输入序列首先通过三个不同的线性变换层，分别得到Query、Key和Value矩阵。这些变换通常是通过全连接层实现的。然后将查询、键和值矩阵分成多个头，每个头具有不同的线性变换参数。对于每个头，都执行一次缩放点积注意力运算。计算查询和键的点积，经过缩放、加上偏置后，使用softmax函数得到注意力权重。这些权重用于加权值矩阵，生成加权和作为每个头的输出。



有了多头注意力的基础，可以进一步实现transformer的encoder；

有一个非常重要的问题是，add&norm层运用的norm使layer\_norm，而不是之前在resnet中使用的batch\_norm,主要有以下原因：

不同句子的长度不同。BatchNorm需要固定长度，而LayerNorm对每个词独立操作，完全不受序列长度影响。

Transformer的核心是自注意力机制，它关注的是一个序列内部元素之间的关系。LayerNorm对每个词的特征进行归一化，确保了模型更关注特征之间的相对关系，而不是特征的绝对数值大小。这非常契合自注意力机制的工作方式。

按照题目要求，随机输入一个数据，随机输入和输出的形状如下，可以看到，两者的形状没有发生改变，可知，输入和输出的形状一致

